ファジング活用の手引き

製品出荷前に機械的に脆弱性をみつけよう

IPA 独立行政法人情報処理推進機構
セキュリティセンター

2013 年 3 月

本書は、以下の URL からダウンロードできます。

「ファジング活用の手引き」

http://www.ipa.go.jp/security/vuln/fuzzing.html

目次

はじめに

ソフトウェア製品において、開発者が認知していない脆弱性（未知の脆弱性）を検出する技術の一つに「ファジング」があります。Microsoft 社など大手企業がソフトウェア製品の開発ライフサイクルにファジングを導入し、製品出荷前の脆弱性検出に活用しています。

本書は、その「ファジング」の概要から実践方法、および製品開発組織におけるファジングの活用方法などをまとめた手引書です。IPA における「脆弱性検出の普及活動」（4 章を参照）の実績に基づき、ファジングの基本的な考え方を理解できるよう本書を構成しています。ソフトウェア製品開発企業におけるファジング活用の一助となれば幸いです。

本書の対象読者

本書の対象読者には、主に組込み機器を始めとしたソフトウェア製品の開発企業における「開発部門」、「品質保証部門（Quality Assuarnce:QA）」の方々を想定しています。

企業によって「品質保証部門」の位置づけが異なることがありますが、本書で「品質保証部門」といった場合、「開発ライフサイクルにおいて出来上がった製品のテストを実施する部門」と位置づけます。

本書の構成

本書は 6 つの章と 1 つの付録で構成しています。

章／付録	章／付録から分かること
1 本書の 3 つのポイント：効果・実績・活用	● ファジングを活用すると、どんな効果を得られるか。
2 ファジングとは	● ファジングとは何か。
3 開発ライフサイクルにおけるファジングの活用	● ソフトウェア製品の開発ライフサイクルにおいて、どの工程でファジングを活用できるのか。
4 IPA におけるファジング実績	● IPA がファジングを実践した結果、どんな実績が得られたか。
5 ファジングの実践	● 実際にどのようにファジングを実践するか。 ● IPA はどのようにファジングを実践したか。
6 ファジング活用に関わる動向	● 企業ではどのようにファジングを活用しているか。 ● 標準化においてファジングはどう位置付けられているか。
付録 A ファジングツールの紹介	● どのようなファジングツールがあるのか。

また「ファジングツールの使い方」や「ファジング結果（パケットキャプチャファイル）を、ファジングツールを使わずに再現する手順」を収録した「ファジング実践資料」を別冊資料として提供しています。

本書における用語

■ **「ソフトウェア製品」**

本書では、「ソフトウェア製品」といった場合、ブロードバンドルーターなどのソフトウェアを搭載した組込み機器も含めます。

■ **「バグ」**

本書では、ソフトウェア製品が仕様通りに動作しなくなる等の問題を「バグ」と定義します。

■ **「脆弱性」**

本書では、ソフトウェア製品を強制的に再起動させてしまう等のセキュリティ上の問題を「脆弱性」と定義します。この定義は、「情報セキュリティ早期警戒パートナーシップ」におけるガイドライン[1]に沿っています。

---

[1] IPA :「情報セキュリティ早期警戒パートナーシップガイドライン」
http://www.ipa.go.jp/security/ciadr/partnership_guide.html 〔2013 年 2 月末日時点〕

1.　本書の 3 つのポイント：効果・実績・活用

■　ファジングの効果

「ファジング」は、脆弱性を検出する検査手法の一つです。コンピュータ性能の向上とともに、近年多数の脆弱性を発見し成果を上げるようになってきています。

ソフトウェア製品の開発ライフサイクルにファジングを導入すると、開発ライフサイクル全体（特にテスト工程）で次の 2 つの効果が得られます。

- バグや脆弱性の低減
- テストの自動化・効率化による労力削減

■　ファジングの実績

IPA が 2011 年 8 月から開始した「脆弱性検出の普及活動」において、6 機種のブロードバンドルーターにファジングしたところ、3 機種で合計 6 件の脆弱性[2]を検出しました。検出したものの中に、インターネットから悪用できるものはありませんでしたが、その一方では複数の脆弱性が検出された機器もありました。

（詳細については「4 IPA におけるファジング実績」を参照してください）

対象機器	脆弱性検出数（*1）	延べ 日数（*2）
ブロードバンドルーターA	2 件	9日
ブロードバンドルーターB	0 件	7日
ブロードバンドルーターC	0 件	9日
ブロードバンドルーターD	3 件	6日
ブロードバンドルーターE	0 件	5日
ブロードバンドルーターF	1 件	9日

（*1）：LAN側でこれらの脆弱性を検出しており、インターネットのWAN側でこれらの脆弱性による事象を再現できませんでした。
（*2）：この延べ日数は単純にファジングを実践した時間だけではなく、検査パターンの特定などの時間も含みます。またファジング担当者はこの作業に専任したわけではありません。

図 2.1-1「脆弱性検出の普及活動」におけるファジング実績

[2] 検出した脆弱性の中にはブロードバンドルーター上で任意のコードを実行できるものが存在する可能性がありますが、IPA では詳細に分析していないため、任意のコードを実行できるか否かは分かりません。

■ ファジングの活用

ファジングの利用方法や効果を知らずに、いきなりソフトウェア製品の開発ライフサイクルでファジングを活用することは難しいでしょう。例えば、ファジングに利用するソフトウェアも多種多様で、商用のものもあればオープンソースソフトウェアやフリーソフトウェア[3]もあります。IPA がオープンソースソフトウェアやフリーソフトウェアを用いて行ったファジングでも、1 日程度で一部の脆弱性を検出することができました（詳細については「3.3 活用のポイント」を参照してください）。

米 Microsoft 社などの大手企業では、実際にソフトウェア製品の開発ライフサイクルにファジングを導入し、出荷前の脆弱性検出に効果をあげています。他にも日本の大手企業の一部でファジングを活用している企業があります。ファジングを行わずに脆弱性を発見する場合、高度な知見と技術力が必要になりますが、ファジングは知見も技術も無かったとしても一定の成果を上げることができます。さらに、人間が作業に関わる部分がデバッグやテストに比べて抑えられるので、脆弱性の修正に掛かる費用を削減できるでしょう。

（詳細については「6.1 企業におけるファジングの活用状況」を参照してください）

[3] 「付録 A. ファジングツールの紹介」で商用製品、オープンソースソフトウェア、フリーソフトウェアのファジングツールを紹介しています。

2. ファジングとは

本章では、ファジングの概要について説明します。

2.1. ファジングとは

「ファジング」とは、検査対象のソフトウェア製品に「ファズ（英名：fuzz)」と呼ばれる問題を引き起こしそうなデータを大量に送り込み、その応答や挙動を監視することで脆弱性を検出する検査手法です。

例えば、あるソフトウェア製品に極端に長い文字列や通常用いないような制御コードなどを送り込み、状態を観察します。その結果、予期せぬ異常動作や異常終了、再起動などが発生した場合、このソフトウェア製品の処理に何らかの問題がある可能性が高いと判断できます。このように、ソフトウェア製品（の開発者）が想定していないデータを入力し、その挙動から脆弱性を見つけ出す検査手法を「ファジング」と言います（図 2.1-1)。
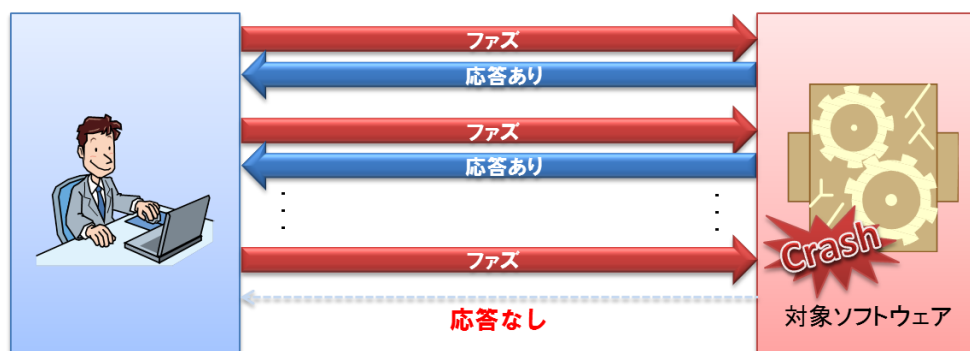
図 2.1-1 ファジングによる脆弱性検出イメージ

2.2. ファズとファジングツール

2.2.1. ファズ

ファズとは、ソフトウェア製品に送り込む「問題を引き起こしそうなデータ」です。ファズのデータ形式はファジング対象によって様々なものがあります。例えば、画像ソフトウェア製品などを対象としたファジングにおいては画像ファイル（ビットマップ、JPEG ファイルなど）がファズであり、ウェブサーバーを対象とした場合では、パケット（HTTP リクエストなど）がファズになります（表 2.2-1)。

表 2.2-1 ファジング対象とファズの形式例

ファジング対象	ファズの形式
画像ソフトなど	ファイル
ウェブサーバ	HTTPリクエスト
ウェブブラウザ	HTTPレスポンス
ネットワーク通信ソフトなど	ネットワークプロトコル(IPプロトコル、TCPプロトコルなど)
CUIプログラムなど	コマンドライン引数、環境変数

ファズのパターンも様々です。例えば、ウェブサーバーに対してファジングを行うファジングツール(ファジングツールについては 2.2.2 節で説明)が生成するファズのパターンには、次のようなものがあります(表 2.2-2)。

表 2.2-2 ファズのパターン例

ファズのパターン	ファズの例	説明
正常値	GET / HTTP/1.0	
リクエストURIに極端に長い文字列を設定	GET AAAAAAAAAAAAAAAAAAA…	スタック・ヒープオーバーフローにつながる可能性がある
リクエストURIに書式文字列を設定	GET %s%s%s%s HTTP/1.0	書式文字列の問題につながる可能性がある
HTTPバージョンに数値の上限を超える値を設定	GET / HTTP/65537	整数オーバーフローにつながる可能性がある

2.2.2. ファジングツール

ファジングを実施するための専用ツールであるファジングツールには、主に「ファズの生成・加工」、「ファズの送信・入力」、「ファジング対象の挙動・死活監視」の 3 つの機能があります。ファジングツールはこれら一連の動作(テストケース)を自動で繰り返し行います。ファジングツールはファザー(英名:fuzzer)とも呼ばれます。
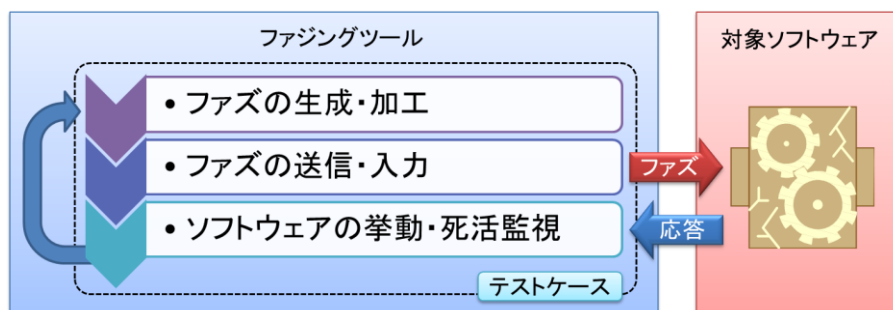
図 2.2-1 ファジングツールの動作イメージ

ファジングツールには様々なものがあり、「生成・加工」されるファズの形式やパターン、「送信・入力」方法、「挙動・死活監視」方法はツールによって異なります(表 2.2-3)。

表 2.2-3 ファジングツールの例

ファジングツール	生成・加工するファズ	送信・入力方法	挙動・死活監視方法
ファイルファジング	ファイル	ソフトウェアの起動引数	ソフトウェアの不正終了
Webサーバーファジング	HTTPリクエスト	ソケット通信	HTTPレスポンスの応答
ブラウザファジング	HTTPレスポンス	ソケット通信	ブラウザの不正終了

また、ファジングツールには商用製品、フリーソフトウェア、オープンソースソフトウェアがありますが、商用製品にはソフトウェアだけでなく、専用機器によって提供されるものも存在します（図 2.2-2）。
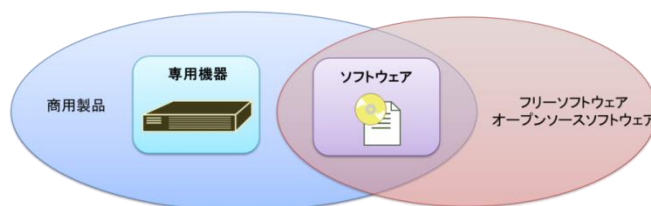
図 2.2-2 ファジングツールの提供形態

2.3. ファジングの特徴・効果・課題

本節ではファジングツールによるファジングの特徴と効果、課題について説明します。本節以降、ファジングと言った場合、ファジングツールを利用することを前提としています。

2.3.1. ファジングの特徴

ファジングには、以下のような特徴が挙げられます。

（1） ブラックボックス検査

- ファジングはソフトウェア製品の内部構造を考慮せず、データの入出力に注目することで外から不具合を見つけるブラックボックス検査です。ソースコードは必要なく、動作するソフトウェア製品があれば実施できます。
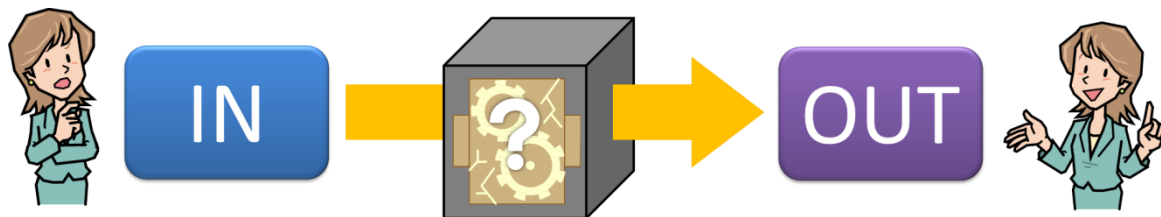
図 2.3-1 ブラックボックス検査

（2）　ブルートフォース

- ファジングは、大量にファズを生成し対象ソフトウェア製品に送り込む、ブルートフォースと呼ばれる総当たり的な検査手法です。このため、手動でファジングを実施することはまれで、ファジングツールにより自動化して行うのが一般的です。
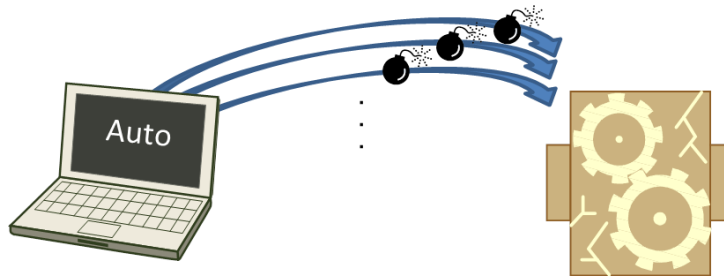
図 2.3-2 ブルートフォース

また、ファジングはシンプルな仕組みのツールを使用して実施するので、複雑な設定を必要としません。そのため、検査対象のソフトウェア製品の知識が少ない人でも比較的導入しやすく、一度使い方を覚えれば誰でも実践できる脆弱性検出手法であると言えます。また、同じテスト手順を自動的に何度も繰り返す仕組みになっているため、検査対象のソフトウェア製品やファジングツールによっては他の作業と並行して実施することも可能です。

一方、ファジングは外から挙動を監視するブラックボックス検査であるため、脆弱性の種類や原因となっている場所を特定できません。そのため、脆弱性を修正するには、別途ソースコードから問題個所を探し出す必要があります。

2.3.2. ファジングの効果

ファジングを実施することにより、以下のような効果を期待できます。

（1）　バグや脆弱性の低減

ファジングはツールを用いて実施するため、何度も繰り返し実施することが比較的容易です。繰り返し実施し問題点の修正を積み重ねることで、バグや脆弱性の低減が期待できます。また、ソフトウェア製品に送り込むファズは、ファジングツールによって機械的に作成するため、ソフトウェア製品の設計者や開発者が想像しえないようなデータによって「想定外」の脆弱性を検出できる可能性があります。設計者や開発者、そしてテスト設計者が想定するテストケースは人為的なものですが、例えば、英語のアルファベットだけを想定しているところに、あらゆる文字コードが混じったデータを機械的に注入すれば、英語のアルファベットでは存在を確認しえなかった脆弱性を発見できる可能性があるのです。

（2）　テストの自動化・効率化による労力削減

多くのファジングツールは人の操作をほとんど必要としないため、比較的楽な工程でバグや脆弱性を検出できると言えます。また、同じような入力データを想定したソフトウェア製品に、ファズを流用したり、再利用したりすることが容易なため、テストの労力や工数の削減にもつながります。

2.3.3. ファジングの課題

ファジングには、どのようなファズをどの程度まで入力すれば十分にテストを行ったと言えるか判断しづらいといった課題があります。

総当たりで組み合わせを試すファジングでは、組み合わせに用いるデータや文字の種類を増やすと、テストの回数が増大し、検査時間が顕著に長くなってしまいます。したがって、組み合わせ数＝テストの回数を減らしつつ、限られた時間内で効果的に脆弱性を発見するための折り合い点を探る必要があるわけですが、その折り合い点をなかなか見出しづらいというのがファジングの課題だと言えます。
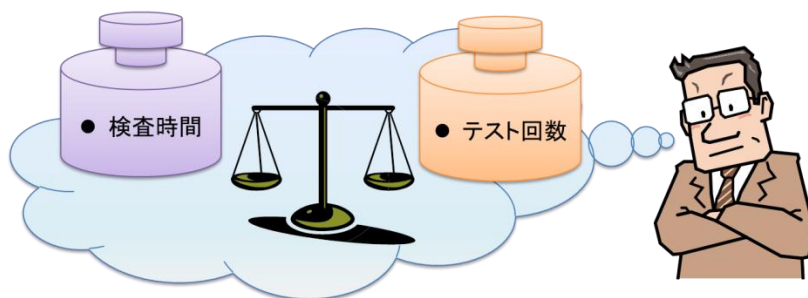
図 2.3-3 ファジングの課題

2012 年 9 月、これらの課題を解消することを目的の一つとして、IPA の検証結果をまとめた「IPA テクニカルウォッチ：製品の品質を確保する『セキュリティテスト』に関するレポート」[4] を公開しました。ファジングの課題を検討するうえで、このレポートもご活用ください。

[4] IPA テクニカルウォッチ：製品の品質を確保する「セキュリティテスト」に関するレポート
～修正費用の低減につながるセキュリティテスト「ファジング」の活用方法とテスト期間に関する考察～
http://www.ipa.go.jp/about/technicalwatch/20120920.html 〔2013 年 2 月末日時点〕

3. 開発ライフサイクルにおけるファジングの活用

本章では、IPA が考える「ソフトウェア製品の開発ライフサイクルにおけるファジングの活用」について説明します。

3.1. 開発ライフサイクルとは

IPA では、ソフトウェア製品の開発ライフサイクルを図 3.1-1 のように考えています。この開発ライフサイクルのうちどこか一工程でセキュリティ対策を実践するのではなく、要件定義から運用、そして廃棄（サービス終了）に至るまですべての工程でセキュリティ対策を実施することが重要だと考えます。
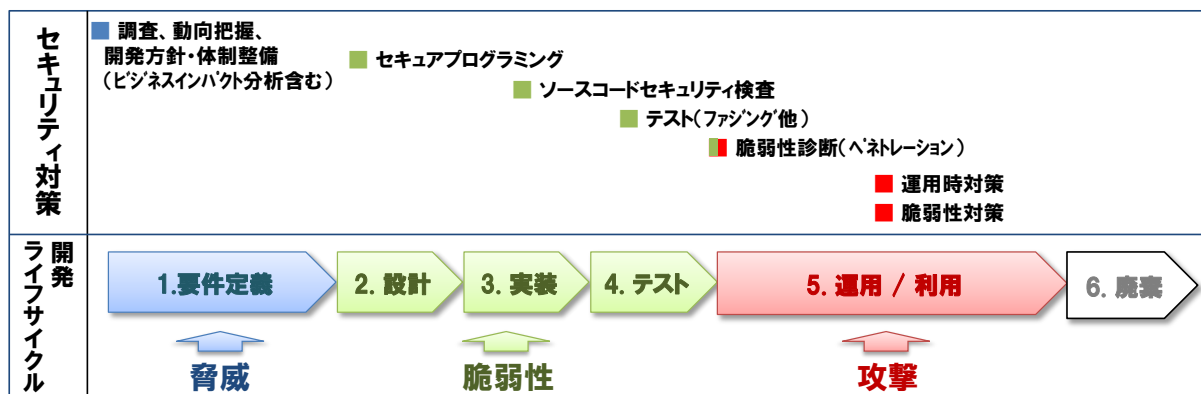
図 3.1-1 ソフトウェア製品の開発ライフサイクル

開発ライフサイクルの各工程で実践すべきセキュリティ対策を下記にまとめます。

（1） 「要件定義」工程

システムの目的や利用形態を明確にし、開発するソフトウェア製品において想定される脅威を洗い出し、それぞれの脅威に対して、ビジネスインパクトや費用対効果を踏まえた対策方針や設計方針を決定します。

（2） 「設計」工程

「要件定義」工程において検討した方針にしたがい、実装すべき機能や取扱うデータ形式などを検討します。実際にソフトウェア製品を運用することを見据えて安全かつ自分達で運用可能な設計をすることが重要です。

また、「セキュアプログラミング」技法の徹底など、脆弱性を作りこまない対策をプログラム実装前に検討します。

（3） 「実装」工程

ソフトウェア製品のソースコードレベルでの対策を行います。プログラミング時の対応はもちろん、ソフトウェア製品の開発者が作ったソースコードに対するチェックも重要です。

（4）　「テスト」工程

「テスト」工程では実行形式のプログラムに対して、動作を確認しながら脆弱性の検査を行います。本書で取り上げている「ファジング」や開発したソフトウェア製品だけではなく OS やミドルウェア等のシステム全体を対象に検査を行う「脆弱性診断」などがあります。

（5）　「運用／利用」工程

脆弱性を悪用する攻撃が日常的に行われていることから、外部からの攻撃の対策として脅威や脆弱性情報の収集、定期的に修正プログラムを適用するなどの脆弱性対策を随時実施します。

この開発ライフサイクルの各工程のうち、特に「実装」、「テスト」の 2 つの工程でファジングを活用できる、と IPA は考えます。

3.2. ファジングを活用する部門

ソフトウェア製品の開発ライフサイクルにおいてファジングを活用する場合、「①ソフトウェアテスト」、「②品質保証」の 2 つの活用目的がある、と IPA は考えます（図 3.2-1）。

図 3.2-1 開発ライフサイクルにおけるファジング活用目的

部門ではありませんが、上記以外には「脆弱性および攻撃手法の研究者」や「攻撃者そのもの」もファジングを利用します。逆に言えば、実装時やテスト時にファジングを行わないということは、攻撃者に対してハンディを負う可能性がある、ということになります。

3.2.1. ソフトウェアテスト

「ソフトウェアテスト」目的で開発部門がファジングを活用する場合、次の 2 つの方法があります。

（1）　ソフトウェアテスト

ソフトウェア製品の開発ライフサイクルにおけるソフトウェアテスト（単体テスト、結合テスト、システムテストなど）で対象機能などにファジングすることでバグや脆弱性を検出します。

（2）　ミドルウェア評価

ソフトウェア製品に組み込むミドルウェアなどの評価において、それらにファジングすることでより脆弱性が少ないミドルウェアを選定します。

3.2.2. 品質保証

「品質保証」目的で品質保証部門がファジングを活用する場合、次の方法があります。

（1）　テスト仕様書への「ファジング」の導入

テスト仕様書に「ファジング」の項目を盛り込み、品質保証検査の一環としてファジングすることで、出荷前にソフトウェア製品の脆弱性を検出します。

3.2.3. その他の目的

■　脆弱性および攻撃手法の研究者

販売されているソフトウェア製品にファジングすることで、攻撃者に悪用される前に未知の脆弱性を検出し、ソフトウェア製品の開発企業に報告します。また、新たな脆弱性や攻撃手法を研究し、その成果をファジングツールに反映させることでファジングによる脆弱性検出の可能性を高めます。

■　攻撃者

販売されているソフトウェア製品にファジングすることで、未知の脆弱性を検出して、それを攻撃に悪用します。

3.2.4. 各部門でファジングを活用するためには

ファジングツールは手法やカバーする文字コードなどが異なるため、ツールによって得意不得意が出てきます。したがって、「①ソフトウェアテスト」、「②品質保証」どちらの活用目的でも、様々なファジングツールをそろえ時間を掛けてファジングすることが最も効果が得られます。しかし、開発工程はそもそも時間が限定的になりがちです。複数のファジングツールを利用するならよけいに効率良くファジングを実践する必要がありますし、そのための体制も必要になってきます。

IPA における「脆弱性検出の普及活動」の実績に基づき、「①ソフトウェアテスト」、「②品質保証」の目的ごとに手軽に実践できるファジング活用のポイントを 3.3 節で取り上げます。

3.3. 活用のポイント

IPA における「脆弱性検出の普及活動」の実績では、オープンソースソフトウェアやフリーソフトウェアのファジングツールを使ったファジングは、1 日程度で完了しています。製品開発企業でまだファジングを導入していない場合、開発部門、品質保証部門において、まず図 3.3-1 のようにファジングを活用するのがよいと IPA は考えます。

■開発部門
オープンソースソフトウェアやフリーソフトウェアのファジングツールのファズを、単体テストや結合テストの入力値として活用する。

■品質保証部門
オープンソースソフトウェアやフリーソフトウェアのファジングツールによるファジングをテスト仕様書の項目に追加する。
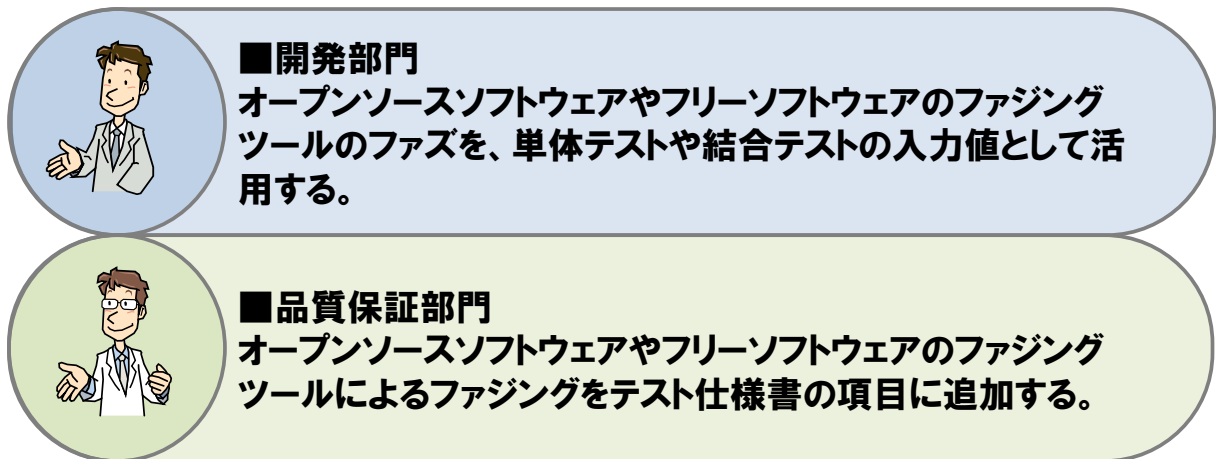
図 3.3-1 製品開発企業によるファジング活用

3.3.1. 開発部門におけるファジングの活用

開発ライフサイクルの後の工程になるに従い、検出した脆弱性を修正するコストは増大します（図 3.3-2）。製品開発における品質保証部門だけではなく、開発部門でもファジングを活用することで、製品出荷後に脆弱性が発見される可能性を低減させることができます。開発部門では、オープンソースソフトウェアやフリーソフトウェアのファジングツールが生成するファズ（詳細は 2.1 節を参照）を単体テストや結合テストなどの入力値として活用するだけでも脆弱性の検出を期待できます。

オープンソースソフトウェアのファズデータの具体的な例として、本書別冊「ファジング実践資料」にファジングツール「Taof」と「Peach」が生成するファズデータを掲載しています。具体的なファズデータを知りたい場合、別途「ファジング実践資料」をご参照ください。
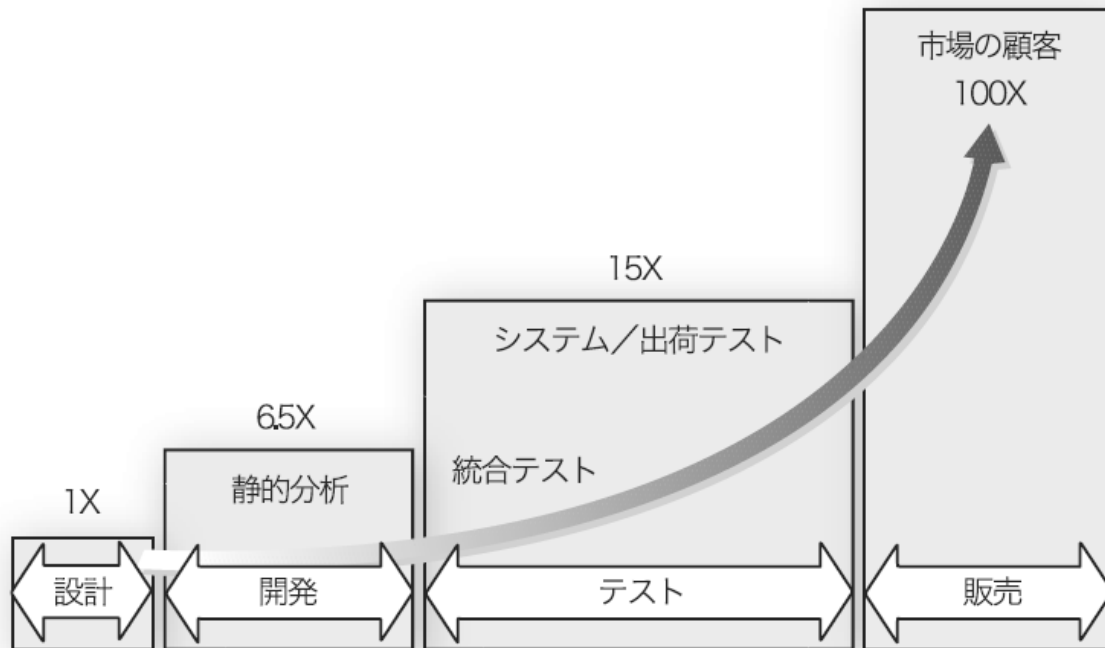
図 3.3-2 開発ライフサイクルにおける脆弱性修正コスト（出典：書籍「ファジング ブルートフォースによる脆弱性発見手法」p.480 図 25-3 SDLC を通じたソフトウェアの欠陥修正コスト）

3.3.2. 品質保証部門におけるファジングの活用

IPA における「脆弱性検出の普及活動」の実績に基づくと、商用製品のファジングツールを活用できれば開発ライフサイクルにおける「テスト」工程でより多くの脆弱性を検出できると考えます。しかしながら、一般的に商用製品は高価です。そのため、ファジングの効果も分からず商用製品の購入に踏み切ることは難しいでしょう。

「脆弱性検出の普及活動」では、オープンソースソフトウェアやフリーソフトウェアのファジングツールを使うと、すぐに脆弱性を検出できてしまった場合がありました。このことから、それらのファジングツールを活用するだけでも、第三者が容易に発見できてしまう脆弱性を「テスト」工程で検出できることが期待できます。

「5. ファジングの実践」にて、ファジングを実践する流れと IPA における「脆弱性検出の普及活用」のファジング実践手順を紹介しています。

「脆弱性検出の普及活動」においては、ブロードバンドルーターに対するファジングを実践しているため、組込み機器を開発している企業の品質保証部門であれば、この実践手順をそのまま活用できると考えます。また組込み機器を開発している企業の品質保証部門でなくとも、ファジング実践のポイントが理解できるよう配慮しています。

4. IPA におけるファジング実績

IPA では、2011 年 8 月からファジングの有効性の実証および普及の促進を目的とした「脆弱性検出の普及活動」に取り組んでいます。この活動では、IPA の担当技術者が実際にファジングを実施し、知見や実績を蓄積しています[5]。なお、IPA で実施したファジングの詳細については、5.2 節を参照してください。
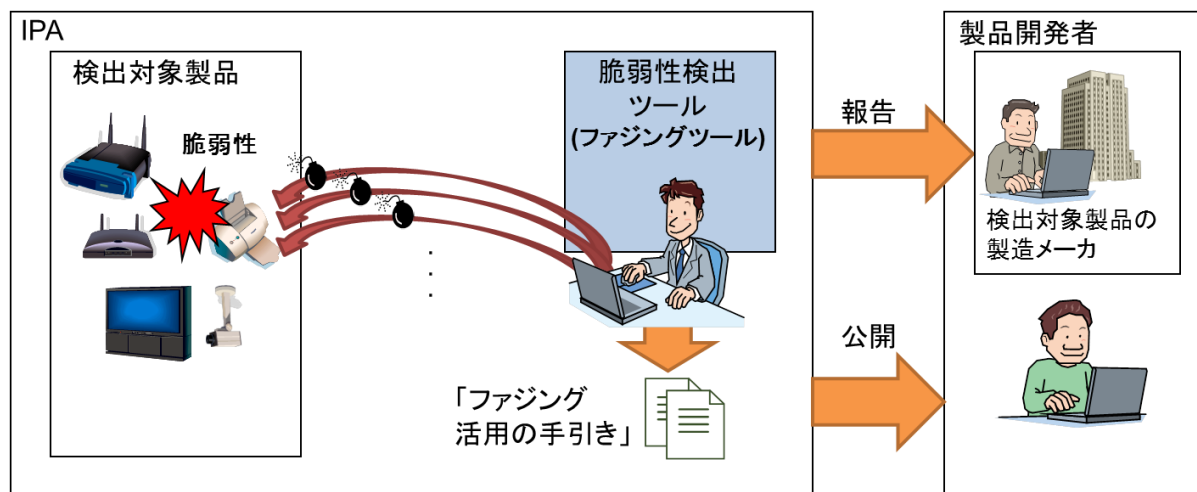
図 3.3-1 「脆弱性検出の普及活動」の概要イメージ

「脆弱性検出の普及活動」において家庭向けネットワーク対応機器にファジングを実施するにあたり、市販製品のうち広く普及しており、扱いやすいネットワーク対応機器であるブロードバンドルーターを選定しました。IPA では 2011 年 12 月～2012 年 2 月まで、ブロードバンドルーター6 機種[6]に対してファジングを実施しました。

1 機種あたりの実施期間は、ファジングによる検出から脆弱性の特定までを含め、最長 2 週間までを目処にしています。また、検査対象機器は 1 機種につき 3 台を用意し、3 人の担当者が、それぞれ異なるファジングツールや異なるテストケースを受け持ち、並行作業にてファジングを実施しました。この際、ファジングの担当者はこの作業に専任したわけでなく、他の業務をしながらファジングを実施しています。
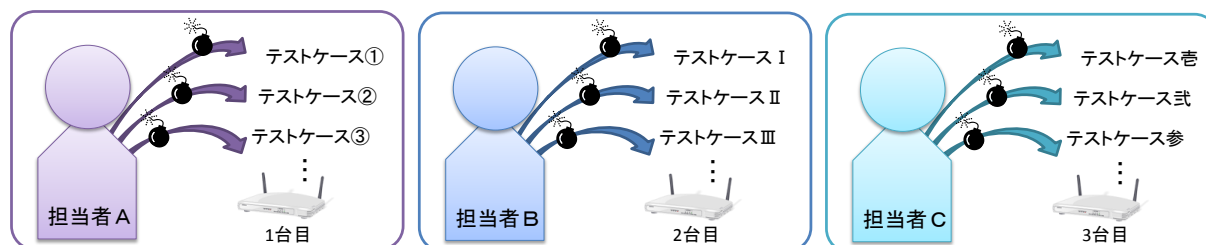
図 3.3-2 同機種3台並行作業によるファジング実施イメージ

[5] IPA :「ソフトウェア製品における脆弱性の減少を目指す「脆弱性検出の普及活動」を開始」
http://www.ipa.go.jp/about/press/20110728.html 〔2013 年 2 月末日時点〕
[6] 2011 年 11 月時点で市販されているブロードバンドルーターのうち、製造メーカーごとに 1 製品ずつ選出しました。

ファジングはブロードバンドルーターの LAN 側、WAN 側の両方を対象に実施しましたが、以下のような機種については WAN 側へのファジングを実施していません。

- WAN 側で Ping への応答などが無く、死活監視が困難なもの
- WAN 側にファジング対象となるサービスがないもの (LAN 側にある HTTP による設定画面への接続サービスなど)

ファジングツールは商用製品 1 種類（ツール①)、オープンソース 2 種類（ツール②、③）の計 3 ツールを利用しました（図 3.3-3)。テストケースは検査対象機器にかかわらず、プロトコルごとに同じパターンを利用しましたが、ファジングにかかる延べ日数は、検査対象機器により日数の開きがありました。これは、検査対象機器によって動作しているサービスやプロトコルの違いがあり、実施するテストケース数に差が発生したためと、検査対象機器の応答速度に違いがあったためです。ただし、IPA で利用したオープンソースソフトウェアのファジングツールによるファジングは、いずれの検査対象機器でも 1 日程度で完了することができました。

ツール	平均テストケース数	平均実施時間
ツール①（商用製品）	116万2,000 件	81時間 00分
ツール②（オープンソース）	13万9,000 件	5時間 15分
ツール③（オープンソース）	1700 件	20分

図 3.3-3 平均テストケース数と実施時間

この活動ではこれまで、3 種の機器で 6 件の脆弱性を検出しています（図 3.3-4)。これらの脆弱性の中には、オープンソースのファジングツールで検出したものや、「機器が強制的に再起動してしまう」、「利用者がインターネットに接続できなくなる」などの事象につながるものもありました。

対象機器	脆弱性検出数（*1）	延べ 日数(*2)
ブロードバンドルーターA	2 件	9日
ブロードバンドルーターB	0 件	7日
ブロードバンドルーターC	0 件	9日
ブロードバンドルーターD	3 件	6日
ブロードバンドルーターE	0 件	5日
ブロードバンドルーターF	1 件	9日

（*1）: LAN側でこれらの脆弱性を検出しており、インターネットのWAN側でこれらの脆弱性による事象を再現できませんでした。
（*2）: この延べ日数は単純にファジングを実践した時間だけではなく、検査パターンの特定などの時間も含みます。またファジング担当者はこの作業に専任したわけではありません。

図 3.3-4 2011 年 12 月～2012 年 2 月までの検出実績

これらの脆弱性はブロードバンドルーターの LAN 側に対するファジングでのみ検出しており、インターネットに接続する WAN 側[7]では、これらの脆弱性を検出したテストケースでファジングしても事象が再現しませんでした（図 3.3-5）。また、これら脆弱性については、機器上で悪意のあるコードを実行できるものが存在する可能性がありますが、コードの実行が出来るか否かについての詳細な分析は IPA では行っていません。

事象	検出したツール		
	①	②	③
ウェブサービスの停止	○	○	○
インターネットへの接続停止	○		
機器の再起動	○		
DHCPサービスの停止	○		
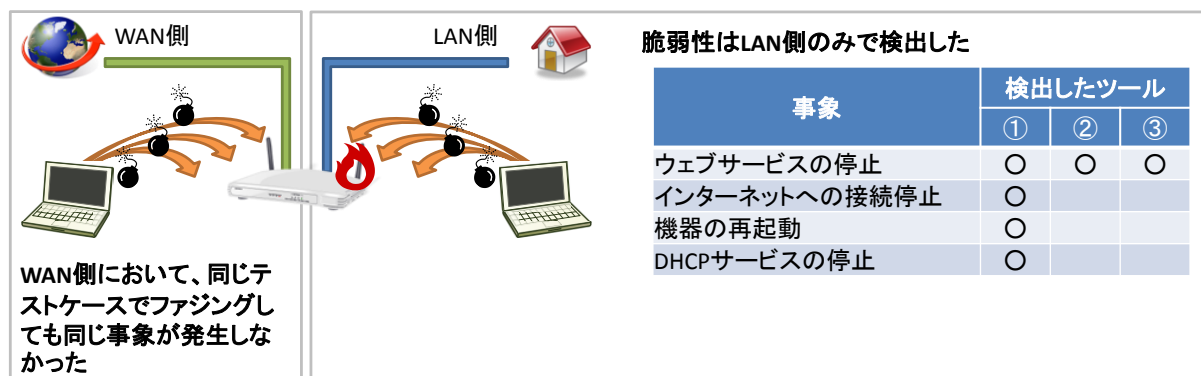
図 3.3-5 IPA が確認した事象と検出したツール

[7] IPA がファジングを実施する場合、ブロードバンドルーターの WAN 側をインターネットには接続せず、閉じたネットワークでファジングを実施しました。

5. ファジングの実践

本章では、ファジングによる脆弱性検出の流れと、「脆弱性検出の普及活動」で実践したファジングについて説明します。

5.1. ファジングによる脆弱性検出の流れ

この節ではファジングによる脆弱性検出の一般的な流れを簡単に説明します。ファジングによる脆弱性検出は、大きく分けて「実行準備」、「検出作業」、「結果分析」の 3 段階でおこないます。
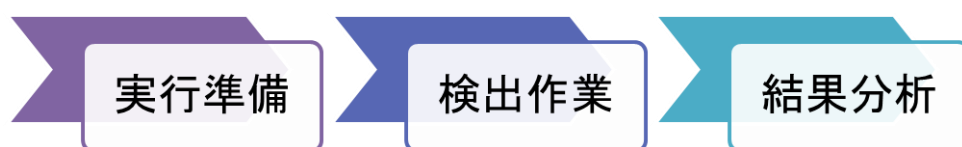
図 5.1-1 ファジングによる脆弱性検出の流れ

（1） 実行準備

「実行準備」では主に、どのようなファジングを実施するのか、どのような動作を監視して脆弱性を検出するかを検討し、目的に合ったファジングツールを選定します。

- ファズと入力先の決定
  対象ソフトウェア製品に対して、どんなファズを、どのように入力するかを決定します。
- 監視方法の決定
  対象ソフトウェア製品のどういった動作を監視し、どうなれば検出とみなすかを決定します。
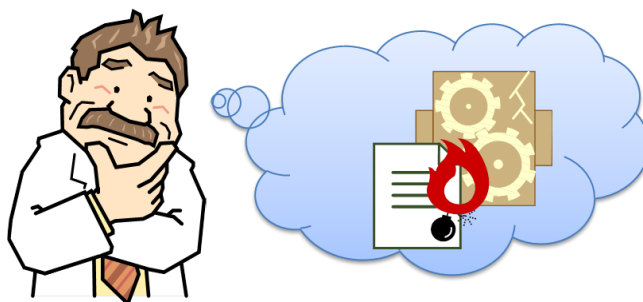
図 5.1-2 実行準備イメージ

対象ソフトウェア製品によって、ファズの入力箇所や形式は異なります。また、ファジングツールによっても、作成されるファズのパターンや入力方法、監視方法は異なります。それぞれの特徴を踏まえ「どこをファジングするか」、「どのファジングツールを使うか」、「どのように監視するか」を検討します。

（2）　検出作業

「検出作業」では実際にファジングを実施し、どのようなファズで問題が発生したかを特定します。

- ファジングの実施
  対象ソフトウェア製品に実際にファジングを実施します。
- ファズの特定
  監視結果（ログファイルやパケットダンプなど）を解析し、どのようなファズで問題が起きたかを特定します。このとき、その問題に再現性はあるかなども調査します。
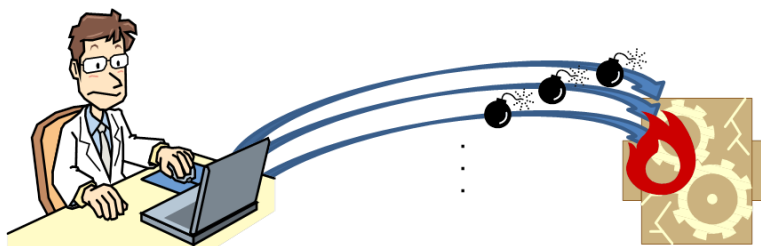
図 5.1-3　検出作業イメージ

検出した問題すべてが脆弱性であるとは限りません。脆弱性と疑わしき問題を検出した場合、どのような条件で問題が発生するかを調査し、脆弱性と判断するための材料を揃えます。

（3）　結果分析

「結果分析」では「検出作業」の結果を分析し、検出した問題が脆弱性であるか判定します。

- 脆弱性の判定
  「検出作業」で特定したファズや、問題が発生する条件などを分析し、検出した問題が脆弱性かどうかを判定します。
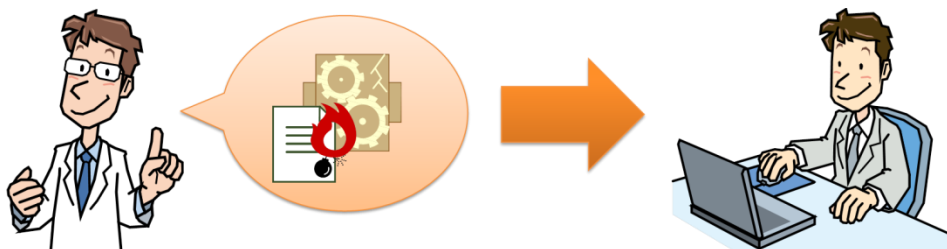
図 5.1-4　結果分析イメージ

結果をまとめ、検査対象の開発者へのフィードバックなどに活用します。

5.2. IPA でのファジング実践

本節では、IPA が家庭向けネットワーク対応機器（ブロードバンドルーター[8]）に対してネットワークを介したファジングを実践した手順を紹介します。この実践手順では、5.1 節で紹介したファジングによる脆弱性検出の流れに沿って、「各作業工程で IPA は具体的には何をしたのか」を説明します。本節を読むことで、ネットワーク対応機器に対する具体的なファジングの流れを理解できます。

製品開発組織などの品質保証部門等でファジングを実践したい場合、本節の実践手順における対象製品などを適宜開発している製品に置き換えることで、組織におけるファジング導入に活用できます。

5.2.1. まえがき

IPA では 2011 年 8 月から「脆弱性検出の普及活動」を開始して、本書執筆時点である 2012 年 2 月末まで大まかに図 5.2-1 のスケジュールで活動しました。
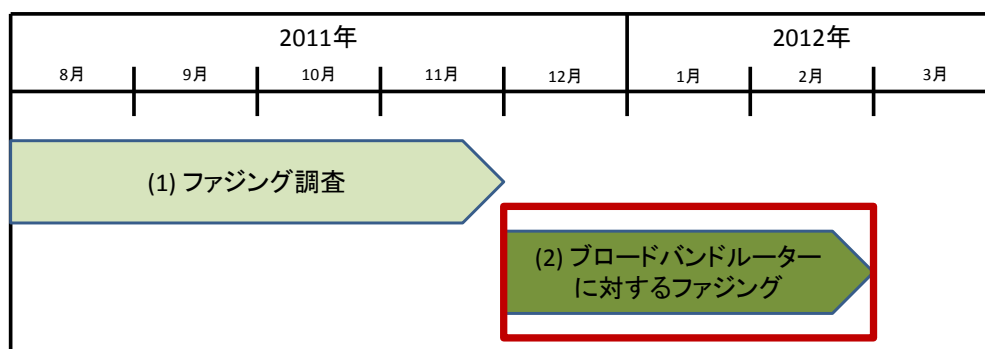
図 5.2-1「脆弱性検出の普及活動」におけるスケジュール

まず 2011 年 8 月から 2011 年 11 月末まで「ファジング調査」ということで、以下のような調査などを実施しました。この「ファジング調査」期間で、ブロードバンドルーターに対するファジングに活用できるファジングツールの選定やファジングを実施するうえで注意することなどを洗い出しました。

- 商用製品のファジングツールやブロードバンドルーターの調達
- オープンソースソフトウェアやフリーソフトウェアのファジングツールの調査[9]
- ファジングツールの試用

2011 年 12 月から 2012 年 2 月末まで「(2) ブロードバンドルーターに対するファジング」を実践しました。5.2.2 節以降、この期間で実践したファジングの実践手順を説明していきます。

[8] 検査対象のブロードバンドルーターには最新ファームウェアを適用し、極力初期設定のまま使用しました。これは、ご家庭での使い方を想定し、すでに開発者が修正した脆弱性を検出しないことを考慮したためです。
[9] IPA が調べたファジングツールに関しては、「付録 A.ファジングツールの紹介」の「オープンソースソフトウェア、フリーソフトウェア」の項をご参照ください。

5.2.2. ファジングの流れ

IPA では、「ファジングで脆弱性を検出できるかを実証する」ことを目的として、市販されているブロードバンドルーターにファジングを実践しました。このファジングの流れは、5.1 節の流れと同様に 3 つの工程に分かれます（図 5.2-2)。「実行準備」、「検出作業」、「結果分析」の流れに沿って、ファジングを実践しました。

実行準備
・ファズと入力先の決定
・監視方法の決定
・ファジング作業計画の決定

検出作業
・ファジングの実施
・ファズの特定

結果分析
・レビュー
・脆弱性の判定
・脆弱性の報告
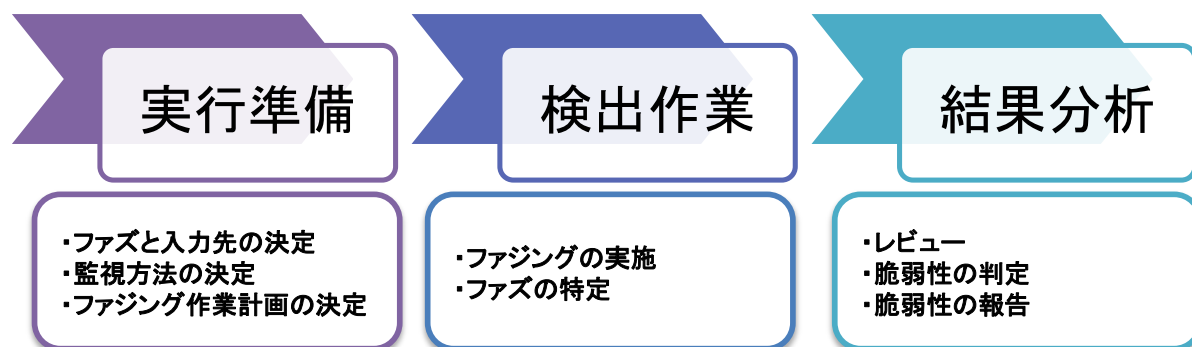
図 5.2-2IPA ファジング実践の流れ

- 実行準備
  ブロードバンドルーターの機能をふまえてファジングツールを決定し、ファジングの作業計画（スケジュールや人員割り当てなど）を立てました。ファジングを実施する準備が整ったら、この工程は完了です。
- 検出作業
  「実行準備」で決定した内容に沿ってファジングを実施し、脆弱性と疑わしき問題を検出します。ファジングが全て終了し、問題の原因となったファズを特定したら、この工程は完了です。
- 結果分析
  「検出作業」で検出した問題が脆弱性かどうか判定します。その判定結果を報告したら、この工程は完了です。

5.2.3. 実行準備

まずファジングを実践する前に、ブロードバンドルーターの機能をふまえてファジングツールを決定し、ファジングの作業計画（スケジュールや人員割り当てなど）を立てました。この工程で実施することを、表 5.2-1 の作業項目に分けて説明します。

表 5.2-1「実行準備」の作業項目と内容

作業項目	作業内容
ファズと入力先の決定	次の内容を決定します。 ● 対象製品のどこをファジングするか ● どのファジングツールを使うか
監視方法の決定	次の内容を決定します。 ● ブロードバンドルーターをどのように監視するか
ファジング作業手順の決定	次の内容などを決定します。 ● ファジングにかける時間 ● ファジングを実践する作業人数

（1） ファズと入力先の決定

IPA では、ブロードバンドルーターに対してファジングを実践するにあたり、商用製品のファジングツール 1 種類とオープンソースソフトウェアのファジングツール 2 種類を使って、ネットワークプロトコルやサービスへのファジングを実施することに決めました。

これらを決定する過程を、「対象製品のどこをファジングするか（入力先の決定)」と「どのファジングツールを使用するか（ファズの決定)」の順に説明します。

■ 対象製品のどこをファジングするか

まず「対象製品のどこをファジングするか」を決めるため、ブロードバンドルーターが取り扱うデータを調査しました。

付属の取扱説明書を確認したり、実際にブロードバンドルーターを動作させた結果、主にTCP/IP や ARP などのネットワークプロトコルのパケット情報を取り扱っていることが分かりました（図 5.2-3）。他にも、ブロードバンドルーター上でウェブサーバーが動作しており、それとやり取りする HTTP のパケット情報も取り扱っていました。

これらの結果をもとに、ブロードバンドルーターで共通している「パケットを処理する機能」（以降、パケット機能）や「ブロードバンドルーター上で動作するサービス」（以降、ネットワークサービス）に対してファジングを実施することに決めました。

このようにソフトウェア製品に対してファジングを実施する場合、その製品の性質を調べてどのようなデータを取り扱っているかを確認する必要があります。
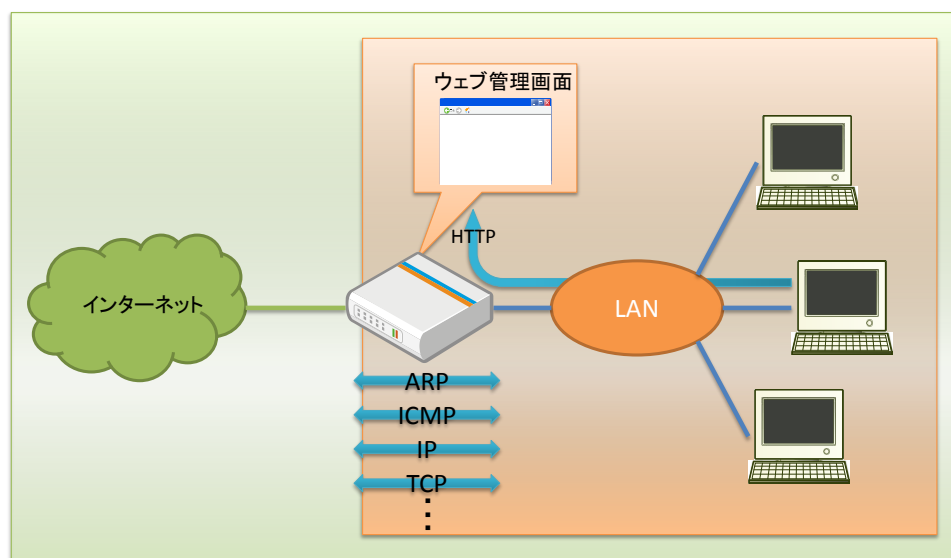
図 5.2-3 ブロードバンドルーターが取り扱うデータ

■ どのファジングツールを使用するか

次にファジングツールを決めます。IPA では、商用製品のファジングツール 1 種類とオープンソースソフトウェア 2 種類の計 3 種のファジングツールを併用することに決めました。

2011 年 8 月から 11 月までの「ファジング調査」で調べたファジングツールのうち、ブロードバンドルーターの「パケット機能」や「ネットワークサービス」に対するファジングを実施できるものとして、商用製品のファジングツール（ツール①）とオープンソースソフトウェアのファジングツール（ツール②、③）を採用しました。各ツールのネットワークプロトコルごとの対応状況は図 5.2-4 の通りです。

ネットワークプロトコル	ツール①	ツール②	ツール③
ARP	○	×	×
DHCP	○	×	×
HTTP	○	○	○
ICMP	○	×	×
IP	○	×	×
TCP	○	×	×
TELNET	○	×	×

○:ファジングを実施できる　×:ファジングを実施できない

図 5.2-4 各ツールのプロトコルごとの対応状況

「ファジング調査」にてファジングツールを試用した結果、対象とするネットワークプロトコルによってファジングの実施時間に差があることが分かりました。

IP や ICMP などによるファジングでは､細工したパケットを 1 つブロードバンドルーターに送るだけで 1 つのテストケースが完了します（1 テストケースにつき 1 パケット）。しかし、TCP や HTTP によるファジングではファジングツールとブロードバンドルーター間で複数のパケットをやり取りする必要があります（1 テストケースにつき複数パケット）。ツール①の TCP によるファジングでは、ファジング開始時の完了予測時間が 100 時間を超える場合がありました。

HTTP におけるファジングは、ツール①だけではなくツール②、③でも実施することが可能でした。ツール②、③はツール①に比べてテストケースが少ないものの、ツール①より短時間でファジングを実施できました。このことをふまえて、IPA では以下のようにファジングツールを使い分けました。

- HTTP 以外のネットワークプロトコルにおけるファジングでは、ツール①を使う。
- HTTP におけるファジングでは、ツール②とツール③を使う。

ファジングする箇所によって、使えるファジングツールが異なります。複数のファジングツールを併用するとファジング時間の短縮につながる場合があります。

（2）　監視方法の決定

次に、脆弱性と疑わしき問題を検出する監視方法を決めます。IPA では、「ファジングツールによる監視」を活用するとともに、「ファジングツールを使わない監視」も併用することとしました。

■　ファジングツールによる監視

IPA で使用したファジングツールには、特定の監視パケットを送信し、その応答の有無などからファジング対象機器の異常動作を検出する監視機能が備わっていました（図 5.2-5）。
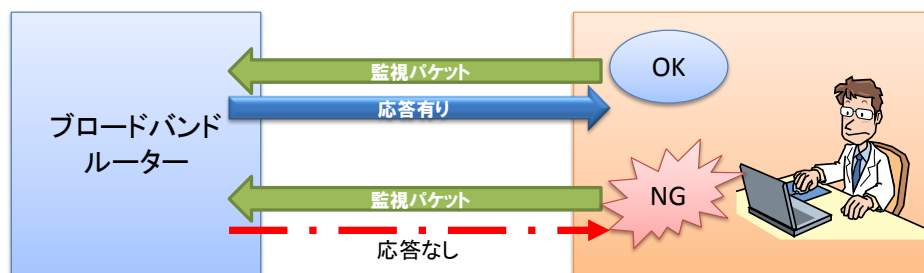
図 5.2-5 ネットワークプロトコルのパケットによるファジングにおける監視イメージ

これらには以下のような監視方法があります（表 5.2-2）。IPA では、基本的にファジングツールに備わっている監視機能を使って、脆弱性と疑わしき問題を検出することに決めました。

表 5.2-2 ネットワークプロトコルのパケットによるファジングにおける監視の例

監視方法	OK	NG
ICMP Echo Request（Ping）	Replyが返ってくる	Replyが返ってこない
TCP通信の確立（TCP SYN パケット送信）	TCP通信が確立する	TCP通信が確立しない

■ ファジングツールを使わない監視方法

しかし、「ファジング調査」にてファジングツールの監視機能だけでは検出できない問題もあることが分かりました。

どのような場合にファジングツールの監視機能だけでは検出できないのか、ファジングによってブロードバンドルーターのウェブサービスが停止した事象を例に説明します。この例では、80/tcp への TCP 通信は確立できるが、ウェブサービスからの応答が無くなるといった事象が発生しました。ファジングツールは「ウェブサービスの 80/tcp と TCP 通信を確立する」という監視を実施していましたが、80/tcp と TCP 通信を確立できてしまったため、異常を検出できませんでした（図 5.2-6）。
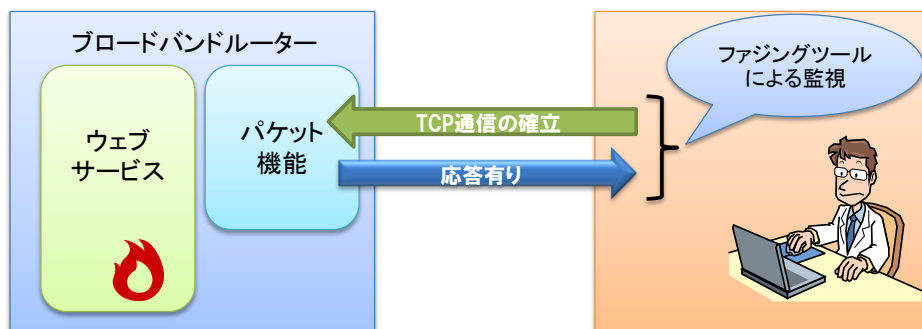
図 5.2-6 ウェブサービス停止時のファジングツールによる監視

しかし、このときウェブブラウザでウェブサービスに接続してみると、ウェブサービスから応答がなくなっており、機器の異常を検出できました。（図 5.2-7）。
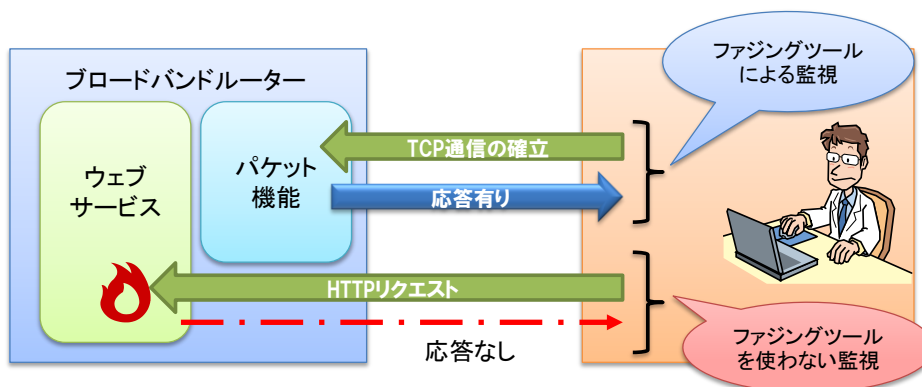
図 5.2-7 ウェブサービス停止時のブラウザによる監視

このことから、IPA ではファジングツールによる監視だけでは検出できない問題もあると判断し、表 5.2-3 の「ファジングツールを使わない監視方法」も併用することにしました。

表 5.2-3 ファジングツールを使わない監視方法

監視方法	内容
ポートの確認	通信ポートの状況を確認します。 通信ポートが稼働していればOK、稼働していなければNGです。
ウェブサービスからの応答確認	ウェブサービスが動作しているか確認します。 ウェブサービスから応答があればOK、応答がない時はNGです。

ファジングツールによる監視だけでは、すべての問題を検出できない場合があります。実際に使用する監視方法がどのような事象を検出できるかを把握して、複数の監視方法を併用して脆弱性と疑わしき問題を検出できるようにしましょう。

（3） ファジング作業計画の決定

事前準備の最後に、IPA ではファジングツールのテストケース数などから「ファジングを実践する作業人数」を決めて、「ファジングにかかる時間[10]」を大まかに算出しました。

「ファジング調査」にてファジングツールを試用したところ、担当者 1 名でファジングを実施すると非常に時間がかかることが分かりました（図 5.2-8 の上部）。

担当者 1 名でファジングを実施する作業を、3 名で分担してファジングを実施すると、大体「2 週間程度」で一通りのファジング作業が完了できる目算が得られました。そこで、ブロードバンドルーター1 機種につき 3 台の機器を用意して、「ファジング担当者 3 名」で並列にファジングを実施することとしました（図 5.2-8 の下部）。

[10] この時間とはファジングを実施するだけではなく、5.2.5 節の「レビュー」や「脆弱性の判定」まで含めた時間を指します。
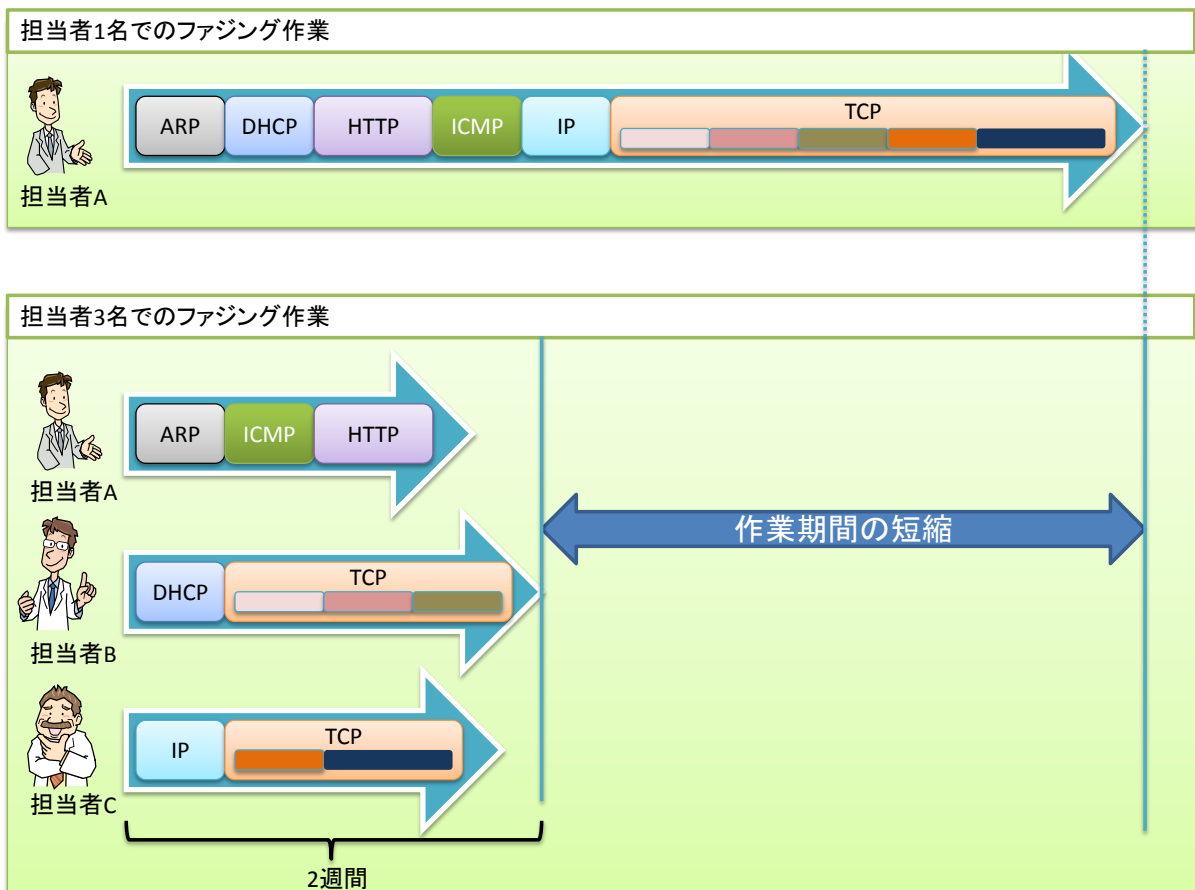
図 5.2-8 ファジング作業の分担

また、並列作業でも効率的に一意のファジングを実施できるように、取得する情報（作業環境、ツール設定、ファジング結果など）や、実施時の確認事項をあらかじめ決めて、ファジング手順書を作成しました。ファジングを実施する前には 1 機種毎に、誰がどのネットワークサービスに対してファジングを実施するかを決めました。

ファジングを実施するために必要なファジングツールや作業計画が決まったら、「実行準備」は完了です。この「実行準備」で決めた項目にしたがって、次の「検出作業」でファジングを実施します。

ポイント!

- **対象ソフトウェア製品によりファズと入力先が異なり、監視方法も変わってきます。準備段階で、対象ソフトウェア製品の特性や機能を理解することが重要です。**
- **ファジングを実施するにあたり、あらかじめ作業計画（作業手順や分担、ファジングかかる時間）を立てましょう。**

5.2.4. 検出作業

「検出作業」では、「実行準備」で決定した内容に沿ってファジングを実施し、脆弱性と疑わしき問題を検出します。この工程で実施することを、表 5.2-4 の作業項目に分けて説明していきます。

表 5.2-4「検出作業」の作業項目と内容

作業項目	作業内容
ファジングの実施	次の作業を実施します。 ● 対象機器にファジングを実施する ● ファジング実践前後で対象機器の動作を確認する
ファズの特定	次の作業を実施します。 ● 検出した問題を引き起こすファズを特定する

（1） ファジングの実施

「事前準備」で決めたファジング作業計画にしたがって、ブロードバンドルーターに対してファジングを実施します。

■ ファジング実施前の動作確認

ファジングを実施する前に、ブロードバンドルーターの「パケット機能」や「ネットワークサービス」が停止していないか確認しました。もしファジングを実施する前からそれらが停止していた場合、ファジングで得られた検出結果を信用できません。

ブロードバンドルーターの動作確認は、「5.2.3.（2）監視方法の決定」で決めた「ファジングツールを使わない監視」を実施しました（表 5.2-5）。もし動作確認の結果が NG となったものがある場合、ブロードバンドルーターを再起動し再度状態を確認しました。

表 5.2-5 ファジング実施前の確認方法

確認方法	内容
ポートの確認	通信ポートの稼働状況を確認するため、ポートスキャンツールを使用します。 ポートスキャンツールは、通信ポートが開いているか閉じているか調査します。 ポートが開いている時はOK、ポートが閉じていたらNGです。
ウェブサービスからの応答確認	ウェブサービスの応答を確認するため、HTTP接続ツールを使用します。 接続ツールでHTTPリクエストを送信して、ウェブサービスからの応答を確認します。 応答メッセージが返ってきたらOK、応答メッセージが返ってこない時はNGです。

■ ファジングの実施

ブロードバンドルーターの動作確認が終わったら、いよいよファジングを実施します。この作業は、ファジングツールごとに必要な情報（ブロードバンドルーターのIPアドレスやMACアドレス、ポート番号など）を設定して、ファジングを開始するだけです。自動的に「事前準備」で決めた監視を実施するようにすれば、ファジングが完了するまで待つだけとなり、ファジング担当者はファジング以外の作業を実施していても構いません。

ファジングの結果、ブロードバンドルーターで以下のような事象を検出した場合、ファジングツールが送信したファズがその事象を引き起こした疑いがあります。

- ブロードバンドルーターが強制的に再起動してしまう。
- ブロードバンドルーターのサービス（ウェブサービスなど）が停止してしまう。
- ブロードバンドルーターから特定のパケット（ARPパケットなど）への応答がなくなる。

またIPAでは、ファジング完了後に「ファジング実施前の動作確認」と同様の方法でブロードバンドルーターの動作を確認しました。このファジング完了後の確認により、ファジングツールによる監視で検出できなかった（見逃した）以下のような事象を検出できました。

- ブロードバンドルーターのウェブサービス（80/tcp）とTCP通信を確立できるが、そのウェブサービスから応答（HTTPレスポンス）がない。

これらの事象を検出したら、この事象を引き起こすファズを次の「ファズの特定」で調べます。

ポイント!

- **対象ソフトウェア製品の動作異常を検出するために、「ファジングツールによる監視方法」と「ファジングツールを使わない監視方法」を併用することは有効です。**

（2） ファズの特定

「ファズの特定」では、ファジング結果から脆弱性と疑わしき問題の原因となったファズ[11]を特定します。

ファジングによって脆弱性と疑わしき問題を検出できた場合、必ずしも直前に送信したファズがそれを引き起こしたとは言えません。特に「ファジングツールによる監視間隔が長い場合」や「ファジングツールの監視方法で問題を検出できなかった場合」、問題を検出する以前のファズが問題を引き起こした可能性が高いと言えます。

■ ファジングツールによる監視間隔が長い場合

「実行準備」で決定した監視方法で問題を検出するわけですが、ファジングツールの性質や設定によってはファズを送信するたびに監視しているわけではありません。

例えば、100 個のファズを送信するたびにファジングツールによる監視を実施する場合を例に説明します（図 5.2-9）。この例で、脆弱性と疑わしき問題を検出した場合、それを引き起こす可能性があるファズは「前回の監視」直後から、「今回検出した監視」までに送信した 100 個のうちのいずれかとなります。これら 100 個のファズのうち、どれが事象を引き起こしたか特定する必要があります。
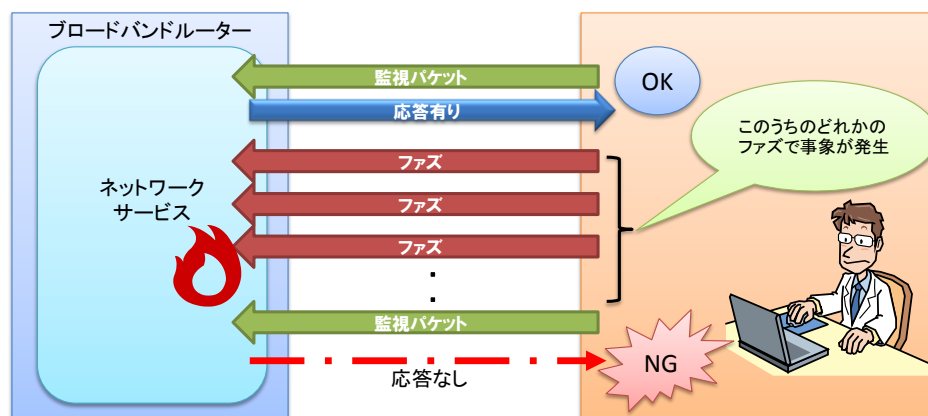
図 5.2-9 ファジングツールによる監視間隔が長いときのイメージ図

[11] ファジングツールはテストケース単位（2.2.2 節）でファジングを実施するため、厳密にはテストケースを特定して、その中に含まれるファズを調べることになります。本節では読みやすさを優先して、テストケースをファズと同義と考えてまとめています。

■ ファジングツールの監視方法で事象を検出できなかった場合

「5.2.3 ■ファジングツールを使わない監視方法」で説明したように、ファジングツールによる監視では問題を検出できない場合があります（図 5.2-10)。事象を引き起こすファズを特定するためには、ファジングツール以外の方法で監視しながら、少しずつファジングを実施する必要があります。
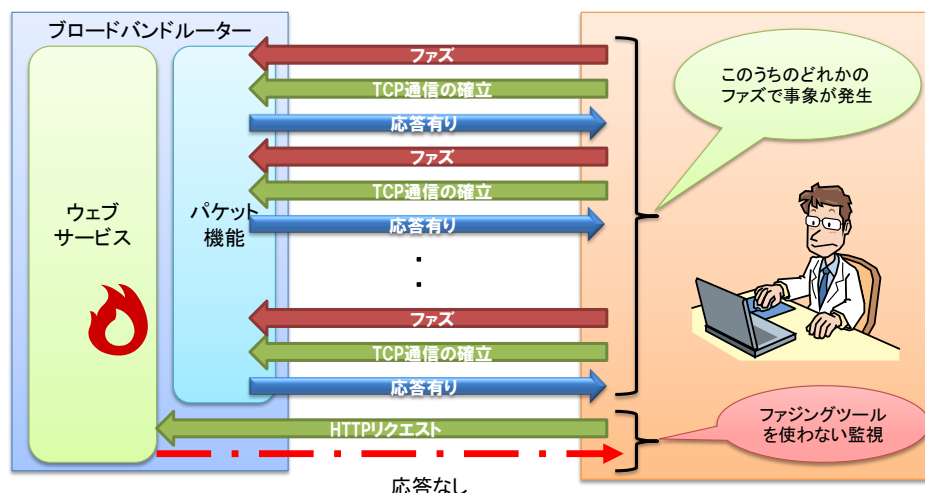
図 5.2-10 ファジングツールの監視方法で事象を検出できなかった場合のイメージ図

以上の 2 つの場合もあることから、ファジングで脆弱性と疑わしき問題を検出したら、実際に問題を引き起こすファズを特定することが重要です。

■ ファズを特定する流れ

ファズを特定する流れは、大きくの 2 つの作業に分けられます。

まず、「ファズを絞り込みながらファジング」を実施します。ここではファジングで検出した事象の直前に送信したファズと対象機器への監視頻度をもとに、ファズの範囲を決めてファジングを実施します。

しかしこの方法では、事象が再現しない場合があります。その場合は「一定量のファズによるファジング」を実施します。

2 つの作業を終えても事象が再現しなかった場合もありますが、そのときはファジングツールによる誤検出の可能性が高いと判断します。

● **ステップ1：ファズを絞り込みながらファジング**

脆弱性と疑わしき問題を検出したファズをもとにして、ファジング時の監視間隔を短くしながらファズを特定していきます。ここでは、100 ファズごとに監視を実施するファジングにおいて、1000 番のテストケース後の監視で脆弱性と疑わしい事象を検出した場合を例に説明します。この例では、956 番目のファズが問題の原因です。

まず 100 ファズごとの監視を実施していたため、脆弱性と疑わしき問題を引き起こしたファズは 901 番から 1000 番の中にある可能性高いと言えます。ここからファズの範囲を絞り込むために、監視間隔を 10 ファズごとに変更して、901 番から 1000 番のファズでファジングを実施します。すると、960 番のファズで同事象を検出しました（図 5.2-11 の絞り込み①)。

さらに監視間隔を 1 ファズごとに変更して、951 番から 960 番のファズでファジングを実施します。すると、956 番のファズで同事象を検出しました（図 5.2-11 の絞り込み②)。

最後に、956 番のファズだけでファジングを実施したときにも、その事象が再現できれば 956 番のファズが問題の原因であると特定できるわけです。
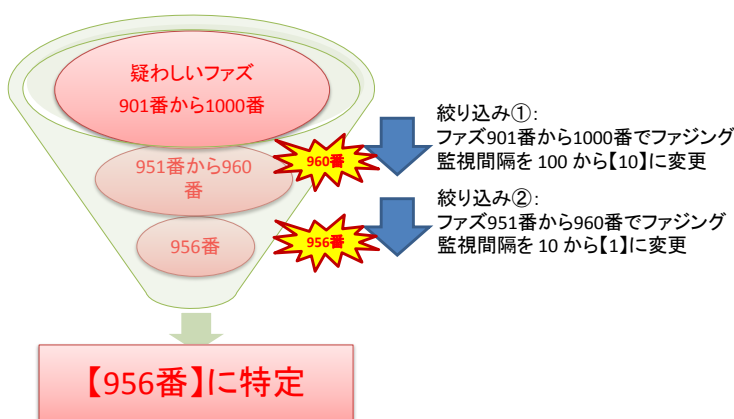
図 5.2-11 ファズを絞り込みながらファジングを実施する例

このステップ 1 を実施すると、特定のファズで問題が発生する場合にそのファズを特定できます（図 5.2-12)。
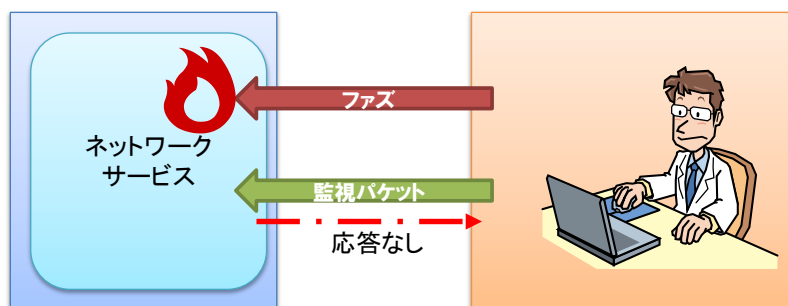
図 5.2-12 特定のファズで脆弱性と疑わしき問題が発生するイメージ図

● **ステップ2：一定量のファズによるファジング**

ステップ１を実施しても検出した事象が再現しなかった場合、1 つのファズでは事象が生じない可能性がでてきます。この場合、一定量のファズにより事象が再現する場合があります（図5.2-13)。
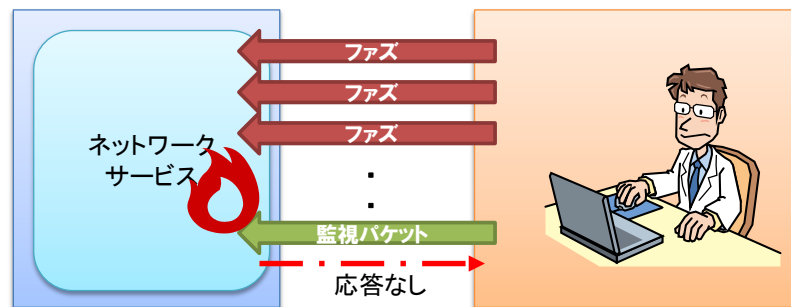
図 5.2-13 一定量のファズで脆弱性と疑わしい事象が発生するイメージ図

問題を検出した直前のファズなどで問題が発生しなかったことから、一定量のファズを送信して問題が発生するか確認します。問題を検出した直前のファズとその前後監視間隔分（100 ファズごとに監視している場合は前後 100 個、計 200 個）のファズで再度ファジングを実施します。その結果、問題が発生したら、一定量のファズにより問題が発生すると判断します。

ステップ１とステップ２の手順で問題が発生しなかった場合は、ファジングツールによる誤検出の可能性を疑います。

「検出作業」では、「実行準備」で決定した内容に沿ってファジングを実施し、脆弱性と疑わしき問題を検出しました。その問題の原因となったファズまで特定できたら、この工程は完了です。

ポイント!

- **脆弱性を疑わしき問題を検出しただけではどのファズでそれが引き起こされたか分かりません。どのようなファズで問題が再現するか、きちんと調べる必要があります。**

5.2.5. 結果分析

ファジング実践の最後である「結果分析」では、「検出作業」で検出したファジング結果に間違いがないかレビューした後、それらの結果が IPA における脆弱性に該当するかを分析します。この工程で分析したファジング結果をブロードバンドルーターの開発者に報告していきます。なお、本書執筆時点である 2012 年 2 月末時点で、検出した脆弱性情報をブロードバンドルーターの開発者に報告しておりません。

（1） レビュー

「検出作業」の結果を、ファジングを実施した担当者以外の別のファジング担当者が確認します。脆弱性情報は開発者にとって重要な情報であるため、ファジングの結果に間違いがあってはいけません。そのためファジング担当者以外の者が客観的にファジング結果に間違いがないか確認します。確認している項目は、取得した情報（実行の記録、実行ログやファジングの結果）が揃っているか、特定したファズにより問題が再現するかなどです。

レビューが終了し、ファジング結果に間違いないことが確認したら、検出したすべての問題点が脆弱性に該当するか判定します。

（2） 脆弱性の判定

レビューしたファジング結果を元に、検出した問題が脆弱性であるかファジング担当者 3 名で判定しました。検出した問題の中には、ブロードバンドルーターの処理能力により生じたと推察できる問題もありました。このような問題については、ブロードバンドルーターの具体的な仕様を知り得ない IPA では脆弱性であるか判断できません。そのため、ブロードバンドルーターが再起動するなど、仕様とは関係なく本来起きてはならない問題を脆弱性と判定しています。なお、脆弱性の判定は「情報セキュリティ早期警戒パートナーシップ」の脆弱性判断に沿ったものとしています。

（3）　脆弱性の報告

IPA では、ファジングで検出した脆弱性情報（脆弱性を引き起こすファズとそれにより生じる事象など）をブロードバンドルーターの開発者に報告する予定です。脆弱性情報を報告する際には、「どのようなファズで脆弱性が生じるのか」をきちんと伝えることはもちろんですが、ファジングツールを使わなくても脆弱性を再現できるように配慮する予定です。「ファジングツールを使わなくとも」としたのは、開発者がファジングツールを所有していない場合や異なるファジングツールを所有している場合が想定できるためです。

ポイント!

- **ファジングツールを使わなくても、開発者がその脆弱性を再現でき、適切に理解できるように配慮した脆弱性情報を提供しましょう。**

■ ファジングツールを使わずに脆弱性を再現する手順

本節で説明したブロードバンドルーターに対するファジングにように、ネットワークを介したファジングでは、脆弱性の原因となるファズを含むファジング結果（パケットキャプチャファイル）があれば、ツールなどを使ってそのファジング結果を再現できる場合があります。IPA では、脆弱性情報を開発者に報告する際には、この再現手順もあわせて報告する予定です。

また、この再現手順を別途手順化して、別冊資料「ファジング実践資料」の「ファジング結果の再現手順」にまとめました。必要に応じて、この別冊資料もご活用ください。

「結果分析」では、「検出作業」で検出したすべての問題が、脆弱性かどうかを判定します。その判定結果を報告したら、この工程は完了です。

5.2.6. IPA でのファジング実践で得られたポイント

これまでのファジング実践を通じて、IPA はファジングを実践する際には、特に次の 2 点が重要であると考えます。

ポイント!

- **対象ソフトウェア製品の特性や機能を理解し、対象ソフトウェア製品の監視方法などを決定しましょう。**
- **検出した脆弱性を開発者に伝える際には、開発者が問題箇所を特定しやすいように、その脆弱性を再現できる情報を整理して伝えましょう。**

この 2 点を念頭においたうえで、本節の IPA でのファジング実践を参考にしていただき、製品検査の手法の一つとしてファジングを活用していただければ幸いです。

6. ファジング活用に関わる動向

6.1. 企業におけるファジングの活用状況

ここでは、企業におけるファジングの活用状況を紹介します。

■ Microsoft

Microsoft 社が実施しているソフトウェア製品開発のセキュリティ保証プロセスである SDL（Security Development Lifecycle : セキュリティ開発ライフサイクル）において、ソフトウェア製品の安全性を高める対策の一つとしてファジングが取り入れられています。特に、Implementation フェーズ[12]（開発ライフサイクルでは「実装」工程に相当）や出荷前の Verification フェーズ[13]（開発ライフサイクルでは「テスト」工程に相当）でファジングが実施されています（図 6.1-1）。
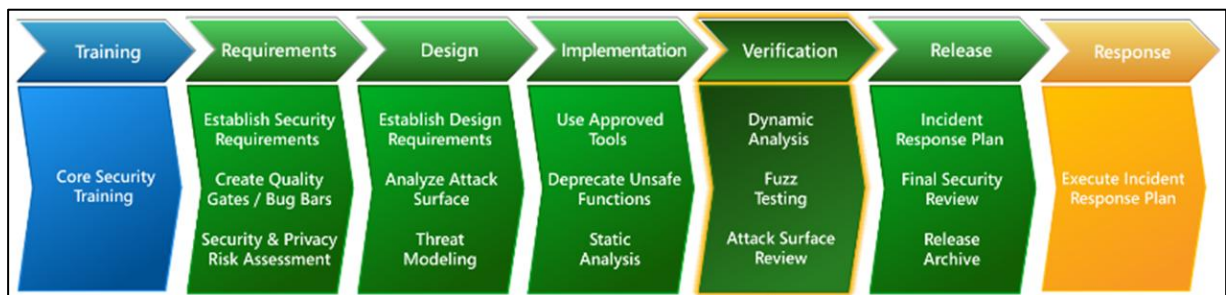
図 6.1-1 Microsoft Security Development Lifecycle（出典 Microsoft:「Software Development Security[14]」より引用）

「Office 2010」のバグや潜在的脆弱性を特定する目的で、実際の開発現場にてファジングが活用された例が紹介されています[15]。このテストでは、「Microsoft Office」の新しい脆弱性を検出するために、300 種を超える形式の Office ファイルが何百万個も用意され、さまざまな方法を駆使したファジングが、毎週何千万回も実施されました。そしてこのテストでは 1800 件の問題点が検出されたと紹介されています[16]。その結果、CERT/CC[17]が発表したファジングによる「Microsoft Office」と「Oracle OpenOffice」のセキュリティ比較によると、前バージョンの Office 製品に比

---

[12] Microsoft :「信頼できるコンピューティングのセキュリティ開発ライフサイクル」
http://msdn.microsoft.com/ja-jp/library/ms995349.aspx#sdl2_topic2_3 〔2013 年 2 月末日時点〕
[13] Microsoft :「SDL Process: Verification」
http://www.microsoft.com/security/sdl/discover/verification.aspx 〔2013 年 2 月末日時点〕
[14] Microsoft :「Software Development Security | Microsoft Security Development Lifecycle (SDL)」
http://www.microsoft.com/security/sdl/default.aspx 〔2013 年 2 月末日時点〕
[15] Microsoft Office 製品開発グループの公式ブログ :「Office のファズ テスト - Microsoft Office 2010 Engineering (日本語訳) - Site Home - TechNet Blogs」
http://blogs.technet.com/b/office2010_jp/archive/2011/06/28/fuzz-testing-in-office.aspx 〔2013 年 2 月末日時点〕
[16] COMPUTERWORLD :「Microsoft runs fuzzing botnet, finds 1,800 Office bugs - Computerworld」
http://www.computerworld.com/s/article/9174539/Microsoft_runs_fuzzing_botnet_finds_1_800_Office_bugs 〔2013 年 2 月末日時点〕
[17]CERT/CC : コンピュータ緊急対応チーム/調整センター（CERT Coordination Center）カーネギーメロン大学ソフトウェア工学研究所が運営するインターネットセキュリティを扱う研究・開発センター

べ脆弱性が減少しています[18]。なお、Microsoft からはファイルファジングツール[19]と、正規表現ファジングツール[20]がフリーソフトウェアとして提供されています。

■ Cisco Systems

Cisco Systems 社でも、脆弱性のリスクを軽減するための開発プロセスである CSLD（Cisco Secure Development Lifecycle：シスコセキュア開発ライフサイクル）の Validate フェーズ（開発ライフサイクルでは「テスト」工程に相当）でファジングが活用されており、商用のファジングツール（Codenomicon Defensics[21]）によって製品の不具合や脆弱性が検出[22]されています（図 6.1-2）。

図 6.1-2　Cisco Secure Development Lifecycle（出典 Cisco Systems：「Cisco Secure Development Lifecycle[23]」より引用）

---

[18] CERT/CC：「CERT/CC Blog: A Security Comparison: Microsoft Office vs. Oracle Openoffice」。
http://www.cert.org/blogs/certcc/2011/04/office_shootout_microsoft_offi.html 〔2013 年 2 月末日時点〕
[19] Microsoft：「Download: SDL MiniFuzz File Fuzzer」
http://www.microsoft.com/en-us/download/details.aspx?id=21769 〔2013 年 2 月末日時点〕
[20] Microsoft：「Download: SDL Regex Fuzzer」
http://www.microsoft.com/en-us/download/details.aspx?id=20095 〔2013 年 2 月末日時点〕
[21] Codenomicon：「DEFENSICS | Codenomicon Defensics」
http://www.codenomicon.com/defensics/ 〔2013 年 2 月末日時点〕
[22] Cisco Systems：「Release Note vA5(1.x), Cisco ACE Application Control Engine Module ; [Cisco Services Modules] - Cisco Systems」
http://www.cisco.com/en/US/docs/interfaces_modules/services_modules/ace/vA5_1_x/release/note/ACE_mod_rn_A51x.html 〔2013 年 2 月末日時点〕
[23] Cisco Systems：「Cisco Secure Development Lifecycle (CSDL) - Cisco Secure Development Lifecycle - Cisco Systems」
http://www.cisco.com/web/about/security/cspo/csdl/index.html 〔2013 年 2 月末日時点〕

■ **日本国内におけるファジングの活用事例**

日本におけるファジングツールの開発企業や販売代理店に IPA がヒアリングを行ったところ、日本でもファジングを導入している企業はあるものの、そういった企業はまだ大手企業に限られることが分かりました。

また、JPCERT コーディネーションセンターと IPA が運営している「情報セキュリティ早期警戒パートナーシップ」においても、ファジングによって検出された脆弱性が届出された実績[24]がありますが、ファジングによって検出された脆弱性の届出はごく一部に過ぎません。

日本では、ソフトウェア製品の開発組織に限らず、脆弱性や攻撃手法の研究者であってもファジングを活用していないのが実情です。

6.2. 組込み制御システム標準化の動向

組込み機器や重要なシステムにおいては、下記のような課題を有しているシステムが数多く存在します。

- 長期間の継続利用（10 年、20 年におよぶ利用等）
- ソフトウェアの更新が困難（セキュリティパッチ等の対応が未確立等）
- 高いセキュリティレベルの要求（人命事故や、事業継続や社会基盤に関わるシステム等）

そうしたシステムにおいては、出荷前に、脆弱性を極力抽出し解消しておくことが重要になります。制御システムがその一つに該当します。生産ラインや輸送系システムや電力・ガス・水道などといった、工業基盤や社会インフラに関わるシステム群です。

制御システムの分野においては、一昨年発覚したウイルス Stuxnet[25]などの事件もあり、セキュリティへの意識やニーズが非常に高まってきています。この制御システムのセキュリティ標準を規定し、それに沿った評価や認証などを確立する動きが進められています。分野共通の基準の策定が進められているものに、IEC62443 と ISASecure[26] EDSA(Embedded Device Security Assurance)が挙げられ、これらは IEC62443 に統合されていく方向にあります。それらでは、装置に内蔵されるシステムのセキュリティ機能、開発プロセス、テスト仕様などが規定されています。評価・認証が先行している ISASecure EDSA では、認証レベルによって、開発プロセスでの「Security Integration Test」でファジングが要件として定められており、また、テスト仕様での「Robustness Test」においてプロトコルファジングの要件が定められています。このテスト仕様は、ISASecure EDSA の認証を取得する場合には必須となっており、ファジングが標準化の流れの中に取り込まれてきています。

---

[24] JVN :「JVN#55714408」http://jvn.jp/jp/JVN55714408/ 〔2013 年 2 月末日時点〕
[25] IPA :「IPA テクニカルウォッチ :『新しいタイプの攻撃』に関するレポート」
http://www.ipa.go.jp/about/technicalwatch/20101217.html 〔2013 年 2 月末日時点〕
[26] ISA Secure : http://www.isasecure.org/ 〔2013 年 2 月末日時点〕

付録 A.ファジングツールの紹介

本付録では、ファジングツールの利用推進を目的としてファジングツールを掲載します。本書2.2 節で説明したように、ファジングツールには商用製品、オープンソースソフトウェア、フリーソフトウェアのものがあります。本付録では、ファジングツールの種類に関わらず掲載します。

本付録に掲載しているファジングツールについては、各製品の開発元企業および開発元のウェブサイトの連絡先にお問い合わせください。特に、オープンソースソフトウェア、フリーソフトウェアを企業で使用する場合、ライセンスおよび利用規約を十分に確認したうえで使用してください。

商用製品

本書の作成にご協力いただいた企業のファジングツールを開発企業名の「あいうえお順（アルファベット順)」に掲載します。

掲載している商用製品のなかには、製品の機能の一つとして「ファジング」機能を搭載しているものがあります。商用製品のファジングツールを網羅的に掲載したいという考えから、それらの製品も掲載しております。

CODENOMICON	
開発元企業	Codenomicon Ltd.
URL	http://www.codenomicon.com/defensics/ （2013 年 2 月末日閲覧）

SPIRENT Communications	
開発元企業	Spirent Communications
URL	http://www.spirent.com/Ethernet_Testing/Software/Studio （2013 年 2 月末日閲覧）
備考	2012 年 4 月に、Spirent Communications 社が Mu Dynamics Inc.社を買収したことで、Spirent Studio 製品に「Mu-8000」等の機能が統合されました。

開発元企業	Wurldtech Security Technologies Inc.
URL	http://www.wurldtech.com/product_services/discover_analyze/achilles_test_platform/ （2013 年 2 月末日閲覧）

IXIA BreakingPoint	
開発元企業	イクシアコミュニケーションズ株式会社
URL	http://www.ixiacom.jp/products/applications/breaking-point （2013 年 2 月末日閲覧）
備考	2012 年 8 月に、イクシア社（本社：米カリフォルニア州カラバサス、NASDAQ:XXIA）が BreakingPointSystems, Inc.社を買収したことで、日本法人のイクシアコミュニケーションズ株式会社が「BreakingPoint FireStorm」等を販売しています。

Raven（レイブン） Product Security Testing Suite	
開発元企業	株式会社フォティーンフォティ技術研究所
URL	http://www.fourteenforty.jp/products/raven/ （2013 年 2 月末日閲覧）

オープンソースソフトウェア、フリーソフトウェア

IPA の「脆弱性検出の普及活動」において動作確認したオープンソースソフトウェア、フリーソフトウェアをツール名の「アルファベット順」に掲載します。各ファジングツールの最新バージョンおよび URL については、2013 年 2 月末日に確認したものを掲載しております。

No	ツール名	最新バージョン	ライセンス	ファジング手法	URL・特記事項
1	<CANVAS> fuzzer	—	記述なし[27]	● ウェブブラウザに対するファジング	http://lcamtuf.coredump.cx/canvas/
2	Browser Fuzzer 3	—	GPL v3	● ウェブブラウザに対するファジング	http://www.aldeid.com/wiki/Bf3
3	BFF (CERT Basic Fuzzing Framework)	2.6	GPL v2	● ファイルによるファジング	http://www.cert.org/download/bff/
4	cross_fuzz	3	記述なし	● ウェブブラウザに対するファジング	http://lcamtuf.coredump.cx/cross_fuzz/ 2011 年 1 月 15 日公開したものが最新バージョンと判断。
5	cssdie	0.7	記述なし	● ウェブブラウザに対するファジング	http://digitaloffense.net/tools/see-ess-ess-die/cssdie.html 上記 URL の HTML コードにおけるコメントにて、最新バージョンを確認した。
6	CSS grammar fuzzer	—	記述なし	● ウェブブラウザに対するファジング	http://www.squarefree.com/css-grammar-fuzzer/ 2009 年 2 月 12 日に公開したものが最新バージョンと判断。
7	DOM-Hanoi	0.2	記述なし	● ウェブブラウザに対するファジング	http://digitaloffense.net/tools/domhanoi/domhanoi.html 上記 URL の HTML コードにおけるコメントにて、最新バージョンを確認した。
8	fuzzball2	0.7	記述なし	● ネットワークを介したファジング	http://www.nologin.org/main.pl?action=codeView&codeId=54 2007 年 6 月以降、アップデートされていない。

[27] 配布されているソフトウェアにライセンスに関する記述を確認できませんでした。

9	fuzz_beacon	4.5.2	3-clause BSD license	● 無線 LAN におけるファジング	http://www.metasploit.com/download/ 「Metasploit Framework」のモジュールの一つである。
10	fuzz_proberesp	4.5.2	3-clause BSD license	● 無線 LAN におけるファジング	http://www.metasploit.com/download/ 「Metasploit Framework」のモジュールの一つである。
11	Hamachi	0.4	記述なし	● ウェブブラウザに対するファジング	http://digitaloffense.net/tools/hamachi/hamachi.html 上記 URL の HTML コードにおけるコメントにて、最新バージョンを確認した。
12	HotFuzz	1.2.0	MIT License	● ファイルによるファジング ● ネットワークを介したファジング ● ウェブアプリケーションに対するファジング	http://hotfuzz.sourceforge.net/ ファジングには「Peach」を使用する。
13	ISIC(IP Stack Integrity Checker)	0.07	BSD License	● ネットワークを介したファジング	http://isic.sourceforge.net/ 2007 年 1 月以降、アップデートされていない。
14	Javascript protocol fuzzer	—	記述なし	● ウェブブラウザに対するファジング	http://www.thespanner.co.uk/2008/06/25/javascript-protocol-fuzzer/
15	JSFuzzer	2.1	GPL v3	● ウェブブラウザに対するファジング	http://code.google.com/p/jsfuzzer/
16	Peach	2.3.8	MIT License	● ファイルによるファジング ● ネットワークを介したファジング ● ウェブアプリケーションに対するファジング	http://peachfuzzer.com/ 2013 年 2 月 7 日に、Peach 3 Release Candidate 1(RC1)が公開されていますが、本資料では 2.3.8 を最新バージョンとして記載した。
17	PROTOS	—	記述なし	● ネットワークを介したファジング	https://www.ee.oulu.fi/research/ouspg/Protos Test Suite ごとの最新バージョンが異なる。
18	ref fuzz	5	Apache License 2.0	● ウェブブラウザに対するファジング	http://code.google.com/p/ref-fuzz/
19	SDL MiniFuzz File Fuzzer	1.5.5.0	記述なし	● ファイルによるファジング	http://www.microsoft.com/en-us/download/details.aspx?id=21769
20	SDL Regex Fuzzer	1.1.0	記述なし	● 正規表現文字列におけるファジング	http://www.microsoft.com/en-us/download/details.aspx?id=20095 関連情報：http://msdn.microsoft.com/ja-jp/magazine/ff646973.aspx

21	Taof(The art of fuzzing)	0.3.2	GPL	●	ネットワークを介したファジング	http://sourceforge.net/projects/taof/ 2007 年 2 月以降、アップデートされていない。
22	XSS tag fuzzer	—	記述なし	●	ウェブブラウザに対するファジング	http://www.thespanner.co.uk/2008/06/18/xss-tag-fuzzer/

更新履歴

更新日	更新内容
2012 年 3 月 27 日	第 1 版 発行。
2012 年 9 月 20 日	第 1 版 第 2 刷発行。 p.4 図 2.1-1 修正。 p.17 図 3.3-4 修正。
2013 年 3 月 18 日	第 1 版 第 3 刷発行。 「2.3.3 ファジングの課題」 加筆。 「付録 A.ファジングツールの紹介」 更新。

著作・制作　独立行政法人情報処理推進機構（IPA）

編集責任　　小林 偉昭

執筆者　　　勝海 直人
　　　　　　岡崎 圭輔
　　　　　　澤田 迅

協力者　　　園田 道夫

　　　　　　甲斐根 功 株式会社日立システムズ
　　　　　　鵜飼 裕司
　　　　　　村上 純一
　　　　　　金野 千里
　　　　　　板橋 博之

　　　　　　渡辺 貴仁　相馬 基邦　田中 里実　谷口 隼祐　千島 幸輔
　　　　　　中西 基裕　山田 祐

協力会社　　株式会社東陽テクニカ
　　　　　　西日本電信電話株式会社
　　　　　　日本コーネット・テクノロジー株式会社
　　　　　　ノックス株式会社
　　　　　　株式会社フォティーンフォティ技術研究所

※独立行政法人情報処理推進機構の職員については所属組織名を省略しました。

ファジング活用の手引き

— 製品出荷前に機械的に脆弱性をみつけよう —

［発　　　行］２０１２年　３月２７日 第1版
　　　　　　　２０１２年　９月２０日 第1版 第2刷
　　　　　　　２０１３年　３月１８日 第1版 第3刷

［著作・制作］独立行政法人情報処理推進機構 セキュリティセンター